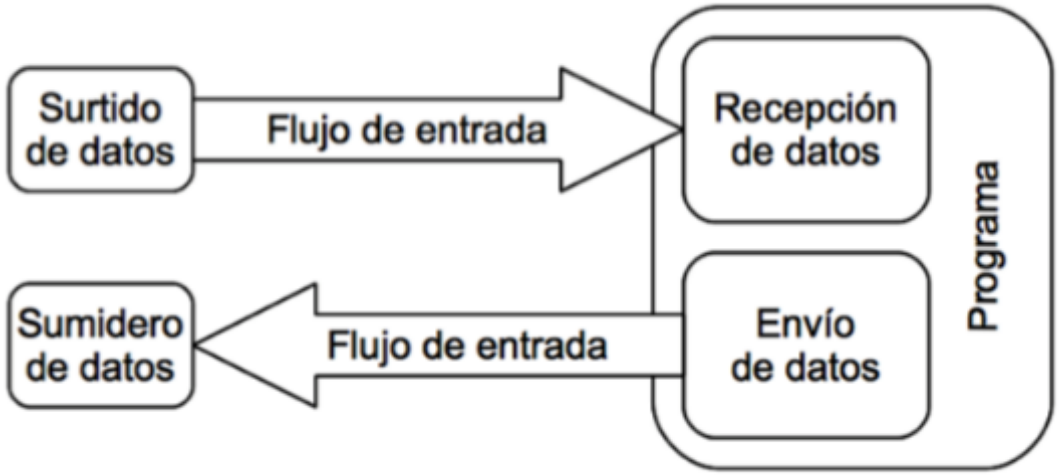
UF1.Persistencia en ficheros

Innovación en Formación Profesional

Entrada y salida datos en java

- Todo lenguaje de programación proporciona una serie de mecanismos para realizar operaciones de entrada y salida de datos. Decimos que los datos son de entrada cuando llegan a nuestra aplicación desde una fuente de datos, y que son de salida cuando nuestra aplicación envía datos a algún sumidero de datos.
- El lenguaje de programación Java nos proporciona un paquete, con una gran cantidad de clases, para poder realizar entrada/salida en nuestras aplicaciones.

Entrada y salida



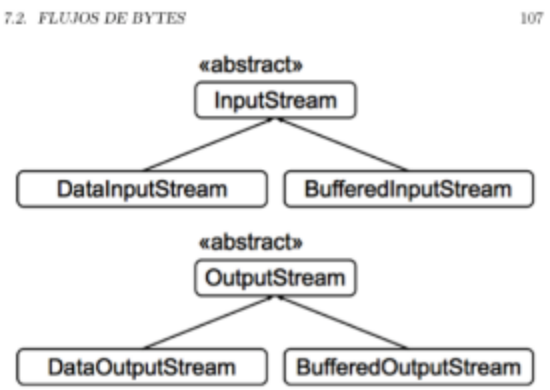
Flujos

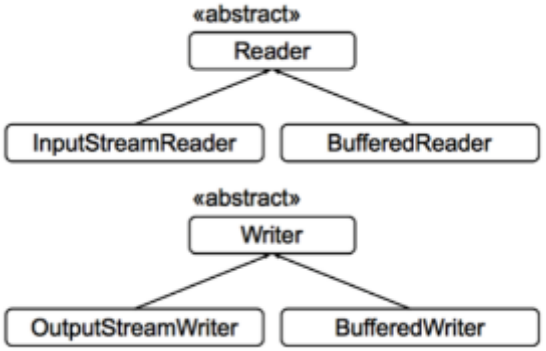
- La potencia de la aproximación de Java a las operaciones de entrada/salida es que Java utiliza un concepto transversal con independencia del dispositivo sobre el que se trabaja. Independientemente de si la salida es hacia un fichero, un Socket o una conexión de Internet, el mecanismo de entrada/salida es el mismo: el uso de Flujos (Streams).

Flujos

- En Java existen dos grandes categorías de flujos, cada una de ellas con sus propias clases para realizar operaciones de entrada salida: los flujos de bytes y los flujos de caracteres. Utilizaremos unos u otros para realizar operaciones de entrada/salida dependiendo de la naturaleza de los datos que recibamos desde una fuente de datos o enviemos a un sumidero de datos.
- **Imput:1,**
- **Buffer: Conjunto**
- FLUJO DE BYTES

FLUJO DE CARACTERES





OPERACIONES SOBRE DIRECTORIOS Y FICHEROS

- Recordad que lar rutas cambian dependiendo del S.O.
- File fichero= new File (“/directorio/ficheros.txt”);→LINUX,OS
- File fichero= new File (“C:\\directorio\\ficheros.txt”);→WINDOWS

Metodo	Funcion
getName()	Devuelve nombre del fichero o directorio
getPath()	Devuevle el camino relativo
getAbsolutePath()	Devuelve el camino absuluto
Exists()	True si existe
canWrite()	True si puede escribir
canRead()	True si se puede leer
isFile()	True si es fichero
isDirectory()	True si es directorio
Lenght()	Longitud en bytes
Mkdir()	Crea un directorio
renameTo(Nuevo nombre)	Renombra
createNewFile()	Crea un fichero vacio
getParent()	Devuelve el nombre del padre, o null

SISTEMA DE FICHEROS Y FLUJOS FICHEROS

- La clase File nos da acceso al sistema de ficheros, con independencia del sistema operativo sobre el que se ejecute. Gracias a esta clase, podemos obtener información tal como la lista de ficheros o directorios bajo un directorio dado, o comprobar si el camino con el que se construye la instancia de File hace referencia a un fichero o a un directorio.
- FIHEROS BINARIOS FICHEROS DE TEXTO

7.5.2.1. Flujos de bytes a ficheros

La clase `FileInputStream` nos permite crear un flujo de lectura hacia un fichero para leer desde él datos de tipo binario. Podemos instanciar esta clase a partir de una referencia a `File` o bien a partir de un `String` que represente el camino hasta el fichero.

De modo análogo, la clase `FileOutputStream` nos permite crear un flujo de escritura hacia un fichero para escribir en él datos de tipo binario. Podemos instanciar esta clase también a partir de una referencia a `File` o bien a partir de un `String` que represente el camino hasta el fichero. En el momento de la creación del flujo podemos indicar si queremos conservar el posible contenido del fichero en el momento de la creación del flujo a través de un argumento de tipo booleano en el constructor.

7.5.2.2. Flujos de caracteres a ficheros

La clase `FileReader` nos permite crear un flujo de lectura hacia un fichero, para leer desde él datos de tipo caracter. De modo análogo al caso de `FileInputStream`, podemos instanciar esta clase s partir de una referencia a `File` o bien a partir de un `String` que represente el camino hasta el fichero.

Finalmente, la clase `FileWriter` nos permite crear un flujo de escritura de caracteres hacia un fichero para escribir en él datos de tipo caracter. De modo análogo a la clase `FileOutputStream`, podemos instanciar esta clase a partir de una referencia `File`, de un `String` que indique el camino al fichero, e indicar en el momento de la creación si el fichero conserva o no el posible contenido.

Como ejemplo del manejo de ficheros, el Listado 7.2 muestra cómo abrir un flujo a un fichero de caracteres para leer desde él línea a línea su contenido y mostrarlo por consola.

FLUJOS DE CARACTERES: FICHEROS DE TEXTO

- Para trabajar con ficheros de texto utilizaremos la clase **Filereader** y **FileWriter**, recordad que deberemos tratar los errores con try-catch.
- LECTURA DE TEXTO CARÁCTER A CARÁCTER

```
import java.io.*;

public class LeerFichTexto {
  public static void main(String[] args) throws IOException {
    File fichero = new
        File("Fichero1.txt");
                //declarar fichero
    FileReader fic = new FileReader(fichero); //crear el flujo de entrada

    //FileReader fic = new FileReader("D:\\ADAT_CLASE\\ANTIGUO\\UNI1\
\LeerFichTexto.java"); //crear el flujo de entrada

    int i;
    while ((i = fic.read()) != -1) //se va leyendo un car·cter
      System.out.println( (char) i + "==>"+ i);

  /*  char b[]= new char[5];
    while ((i = fic.read(b)) != -1)
                System.out.println(b);
*/


    fic.close(); //cerrar fichero



  }
}
```

- ESCRITURA DE TEXTO CARÁCTER A CARACTER

```
import java.io.*;

public class EscribirFichTexto {
  public static void main(String[] args) throws IOException {
   File fichero = new File("FichTexto.txt");//declara fichero
   FileWriter fic = new FileWriter(fichero); //crear el flujo de salida
   String cadena ="Esto es una prueba con FileWriter";
   char[] cad = cadena.toCharArray();//convierte un String en array de caracteres

   for(int i=0; i<cad.length; i++)
           fic.write(cad[i]);  //se va escribiendo un car·cter

   fic.append('*'); //aÒado al final un *
   fic.write(cad);//escribir un array de caracteres
   String c="\n*esto es lo ultimo*";
   fic.write(c);//escribir un String

   String prov[] = {"Albacete","Avila","Badajoz",
                    "C·ceres","Huelva","JaÈn",
           "Madrid","Segovia","Soria","Toledo",
           "Valladolid","Zamora"};

   fic.write("\n");
   for(int i=0; i<prov.length; i++) {
              fic.write(prov[i]);
              fic.write("\n");
   }
   fic.close();    //cerrar fichero
  }
}
```

- LECTURA DE TEXTO LINEA A LINEA

```
import java.io.*;
public class LeerFichTextoBuf {
  public static void main(String[] args) {
  try{
          File fic = new File("FichTexto.txt");//declara fichero
      BufferedReader fichero = new BufferedReader(
                        new FileReader(fic));


      String linea;
      while((linea = fichero.readLine())!=null)
        System.out.println(linea);
      fichero.close();
        }
        catch (FileNotFoundException fn ){
               System.out.println("No se encuentra el fichero");}
        catch (IOException io) {
               System.out.println("Error de E/S ");}

  }
}
```

- ESCRITURA DE TEXTO LINEA A LINEA

```
import java.io.*;
public class EscribirFichTextoBuf {
  public static void main(String[] args) {
  try{
      BufferedWriter fichero = new BufferedWriter
                  (new FileWriter("FichTexto1.txt"));
      for (int i=1; i<11; i++){
            fichero.write("Fila numero: "+i); //escribe una lÌnea
            fichero.newLine(); //escribe un salto de lÌnea
      }
      fichero.close();
      }
        catch (FileNotFoundException fn ){
               System.out.println("No se encuentra el fichero");}
        catch (IOException io) {
               System.out.println("Error de E/S ");}
  }
}
```

FLUJOS DE BYTES: FICHEROS BINARIOS

- Los ficheros binarios almacenan secuencias de digitos binarios que no son legibles por el usuarios, tienen la ventaja que ocupan menos espacio. En java se utiliza **FileImputStream** y **FileoutpuStrem**
- ESCRITURA BYTES EN FICHERO y LEER

```
import java.io.*;
public class EscribirFichBytes {
  public static void main(String[] args) throws IOException {
   File fichero = new File("FichBytes.dat");//declara fichero
   //crea flujo de salida hacia el fichero
   FileOutputStream fileout = new FileOutputStream(fichero,true);
   //crea flujo de entrada
   FileInputStream filein = new FileInputStream(fichero);
   int i;

   for (i=1; i<100; i++)
       fileout.write(i); //escribe datos en el flujo de salida
   fileout.close(); //cerrar stream de salida

   //visualizar los datos del fichero
   while ((i = filein.read()) != -1) //lee datos del flujo de entrada
            System.out.println(i);
   filein.close();  //cerrar stream de entrada
  }
}
```

FLUJOS DE BYTES: FICHEROS BINARIOS CON BYTE

```
FileOutputStream escribir = new FileOutputStream("fichero.dat");

                                        System.out.print("\n"+"Introduce el nombre de usuario: ");
                                            String usuario="";
                                            usuario = sc.next();
                                        System.out.print("Introduce la contrasena: ");
                                            String password="";
                                            password = sc.next();
                                        String esp = "    ";
                                        byte usr[]=usuario.getBytes();
                                        byte espa[]=esp.getBytes();
                                        byte passw[]=password.getBytes();

                                        escribir.write(usr);
                                        escribir.write(espa);
                                        escribir.write(passw);

                                        System.out.println("\n"+"[Datos almacenados correctamente.]");

                                        System.out.println("\n"+"Leyendo las credenciales... [Usuario] y [Contrasena]");
                                        FileInputStream re = new FileInputStream("fichero.dat");
                                            int valor = re.read();
                                            while(valor!=-1){
                                                System.out.print((char)valor);
                                                valor = re.read();
                                            }
```

SERIALIZACIÓN

- Para que un programa java pueda almacenar o leer un objeto, la clase necesita ser serializable. Para ello debe implementar la interfaz Serializable.

- Ejemplo:
- Public class Persona implements Serializable
- {
- ------
- ------
- -----
- }

SERIALIZACIÓN

- También podemos guardar objetos enteros en un fichero. Este proceso se denomina serialización. Para ello utilizaremos la clase **readObject**() y **writeObject**()
- LECTURA DE OBJETOS EN FICHERO

```
import java.io.*;

public class LeerFichObject {
        public static void main(String[] args) throws IOException, ClassNotFoundException {
                Persona persona; // defino la variable persona
                File fichero = new File("FichPersona.dat");
                ObjectInputStream dataIS = new ObjectInputStream(new
FileInputStream(fichero));

                int i = 1;
                try {
                        while (true) { // lectura del fichero
                                persona = (Persona) dataIS.readObject(); // leer una Persona
                                System.out.print(i + "=>");
                                i++;
                                System.out.printf("Nombre: %s, edad: %d %n",
                                                persona.getNombre(),persona.getEdad());

                        }
                } catch (EOFException eo) {
                        System.out.println("FIN DE LECTURA.");
                } catch (StreamCorruptedException x) {
                }

                dataIS.close(); // cerrar stream de entrada
        }
}
```

- ESCRITURA DE OBJETOS EN FICHERO

```
import java.io.*;

public class EscribirFichObject {
  public static void main(String[] args) throws IOException {

   Persona persona;//defino variable persona

   File fichero = new File("FichPersona.dat");//declara el fichero
   FileOutputStream fileout = new FileOutputStream(fichero,true);  //crea el flujo de
salida
    //conecta el flujo de bytes al flujo de datos
   ObjectOutputStream dataOS = new ObjectOutputStream(fileout);

   String nombres[] = {"Ana","Luis Miguel","Alicia","Pedro","Manuel","AndrÈs",
                       "Julio","Antonio","MarÌa Jes´s"};

   int edades[] = {14,15,13,15,16,12,16,14,13};
   System.out.println("GRABO LOS DATOS DE PERSONA.");
   for (int i=0;i<edades.length; i++){ //recorro los arrays
      persona= new Persona(nombres[i],edades[i]); //creo la persona
          dataOS.writeObject(persona); //escribo la persona en el fichero
          System.out.println("GRABO LOS DATOS DE PERSONA.");
   }
   dataOS.close();  //cerrar stream de salida
   }
}
```

Gracias por su atención

Jorge Peris

Innovación en Formación Profesional